Requirements Models for System Safety and Security

Connie Heitmeyer
Naval Research Laboratory

International Summer School
Marktoberdorf
August 2010

OVERVIEW

- Introduction to the Requirements Problem
- Four Variable Model and SCR
  - Formal Requirements Model
  - Analysis of Requirements Models
- Verifying Source Code for Security Properties: A Practical Application
- An incremental, model-based method for developing critical software
  - Example applying the method to fault-tolerance

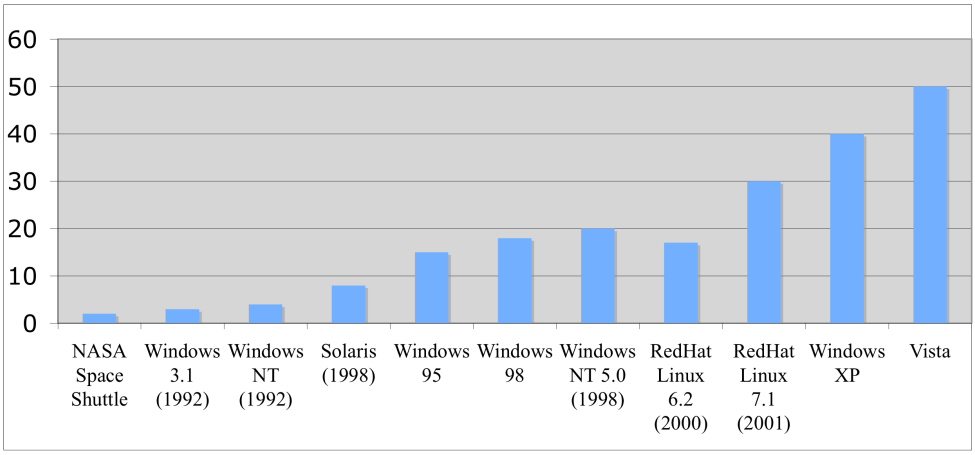
THE CURRENT STATE OF SOFTWARE

Dramatic Increase in the Size and Complexity of Software

Growth of software in military aircraft

Year	Aircraft	% of Pilot Functions
1960	F-4	8%
1982	F-16	45%
2000	F-22	80%

Millions of lines of code in systems



Of the Major Components of Systems, SOFTWARE Is the Most Problematic

DOD … spends about 40% of its Research, Development, Test, and Evaluation budget on **software**—$21B for fiscal year 2003 ... DOD and industry experience indicates that about $8B (40 percent) of that amount may be **spent on reworking software because of quality-related issues.**

U.S. GAO, "Defense Acquisitions," 2004 Report to Committee on Armed Services U.S. Senate

Details Emerge On Army's Failed NLOS-LS Missile

In testimony before lawmakers yesterday, David Duma…detailed failings of the Army Non-Line of Sight Launch System (NLOS-LS). During most recent tests in February, new navigation software caused six of seven total system aborts.

Defense Tech, April 16, 2010

A U.S. soldier in Afghanistan used a Precision Lightweight GPS Receiver to set coordinates for an air strike. Seeing that the "battery low" warning light was on, he changed the battery, then pressed "Fire." The device was designed, on starting or resuming operation after a battery change, to initialize the coordinate variables to its own location…

The soldier and three comrades were killed in the incident.

"'Friendly Fire' Deaths Traced to Dead Battery: Taliban Targeted, but US Forces Killed," *Wash. Post*, 22 Mar. 2002

MANY OTHER EXAMPLES OF SOFTWARE FAILURES*

Aviation: Many recent incidents illustrate risks due to software

- 1997 crash of Korean 747 in Guam: 200 deaths that probably could have been avoided if altitude warning system had been configured correctly
- 2004 air traffic control outage in Palmdale CA disrupted 800 flights:
  - Prevented any voice communications between controllers and aircraft
  - Aircraft violated minimum separation distance five times; only because of airline collision avoidance systems that no collisions occurred

Medical Devices: Failures in medical devices can be lethal

- From 1990-2000, safety recalls of pacemakers and implantable defibrillators due to software problems affected over 200,000 devices
- From 1992-98, an FDA study found that 242 of 3,140 device recalls were due to faulty software (80%+ due to defects introd'd in maintenance)

Voting: Much voting software is insecure and unreliable

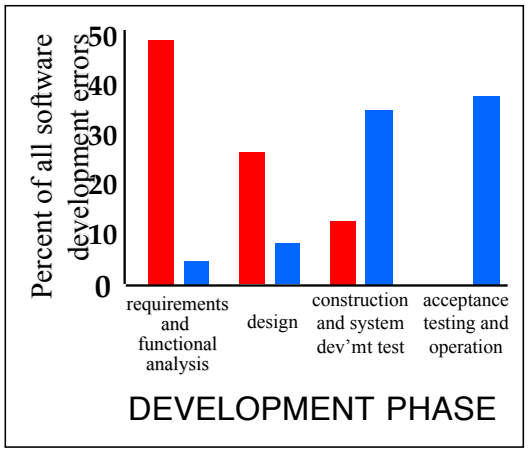
- In 2006 in Sarasota FL, election was decided by a margin of 363 votes
  - But 18,000+ ballots cast on electronic voting machines did not register a vote

*Jackson et al., Software for Dependable Systems: Sufficient Evidence?, NAS, 2007.

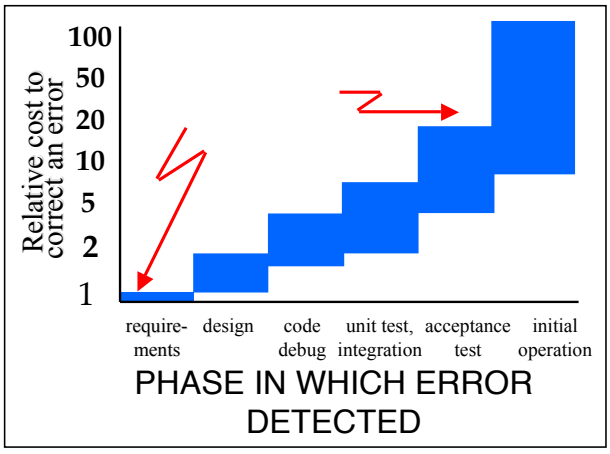
THE IMPORTANCE OF HIGH-QUALITY REQUIREMENTS

REQUIREMENTS ERRORS ARE PERVASIVE AND COSTLY

1. The majority of software errors are introduced early in software development



2. The later software errors are detected, the more costly they are to correct



KEY:
- error introduced
- error detected

THE REQUIREMENTS PROBLEM: FORMAL METHODS RESEARCHERS*

In spite of…advancements…, the biggest problem in software engineering [is] the bridging of the 'gap' between the intent captured in requirements and expressed at a high level, and the detailed encoding of this intent in the code. There are no good tools, either mental or mechanical, that allow comprehension of large programs, and provide a mapping between how…parts of the code work together to satisfy the requirements…

Sriram Rajamani, "Software is more than code"

A final difficulty encountered in modeling is the frequent lack of good requirement documents associated with the project. Most of the time, industrial requirement documents are either almost nonexistent or far too verbose. Usually they have to be rewritten before serious modeling starts.

Jean-Raymond Abrial, "Theory becoming practice"

There is general consensus that the most significant problems in software development are due to inadequate requirements, especially where these concern what one component or subsystem may expect of another.

John Rushby, "Automated formal methods enter the mainstream"

* *Journal of Universal Computer Science, May 2007*

WRITING GOOD REQUIREMENTS IS VERY DIFFICULT

The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements…No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.

Fred Brooks
"No Silver Bullet: Essence and Accidents of Software Eng.," Computer, 1987

GOOD REQUIREMENTS: PAYOFF

A solid foundation for demonstrating high assurance that the implementation satisfies critical system properties, e.g.,

- Safety
- Security
- Timing
- Fault-tolerance
- Service

THE REQUIREMENTS PROBLEM

REQUIREMENTS GOAL

- Requirements Goal: Specify the set of all acceptable implementations
  - Avoid underspecification (completeness)
  - Avoid overspecification (freedom from implementation bias)
- Target: High Assurance Systems
  - Complex, often large, embedded systems
    - » Avionics systems, medical systems, control systems
    - » Expensive and difficult to build correctly
  - Software is safety-critical/security-critical/…
    - » Small errors -> BIG PROBLEMS!
    - » High cost of failure

THE FORMAL METHODS DILEMMA

- Standard approaches (e.g., prose specs) lack sufficient rigor to meet high-assurance goals
- Formal requirements methods have desired technical attributes but viewed as impractical for large, complex systems
  - Capability for unambiguous specification, precision, testability, and analyzability
  - Industry wants *practical* methods
    - » Specs should be easy to read and write, should not require mathematical sophistication, method must scale
    - » Concern for real-world issues of fuzzy or changing requirements
    - » Concern for fit with current development method
    - » **Adds up to perceived cost/schedule risk**
- Implication: Need for "Practical" Formal Methods

FOUR VARIABLE MODEL

FOUR VARIABLE MODEL*

- Generalized approach to requirements in A-7 requirements document (Heninger, TSE, 1980)
- Requirements are specified in two stages
  - Ideal system behavior (I.e., normal behavior)
  - Real system behavior
- The required system behavior is expressed in terms of quantities in its environment
  - The application dictates the environmental quantities of interest
  - The relevant env. quantities are represented as mathematical variables called *environmental variables*

Two of the Four Variables of the FVM

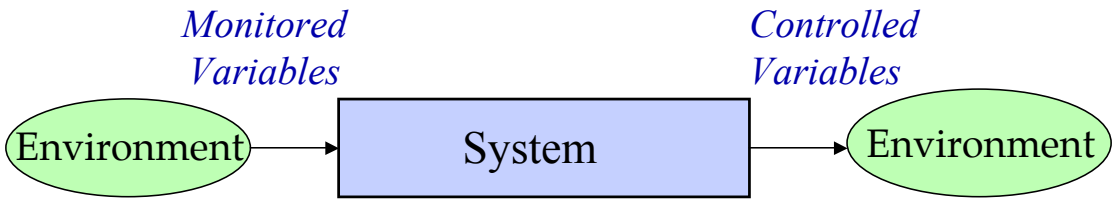
(1) Monitored Variables: env. quantities that the system monitors (e.g., temperature, pressure, altitude)

(2) Controlled Variables: env. quantities that the system controls (e.g., displayed value, throttle)

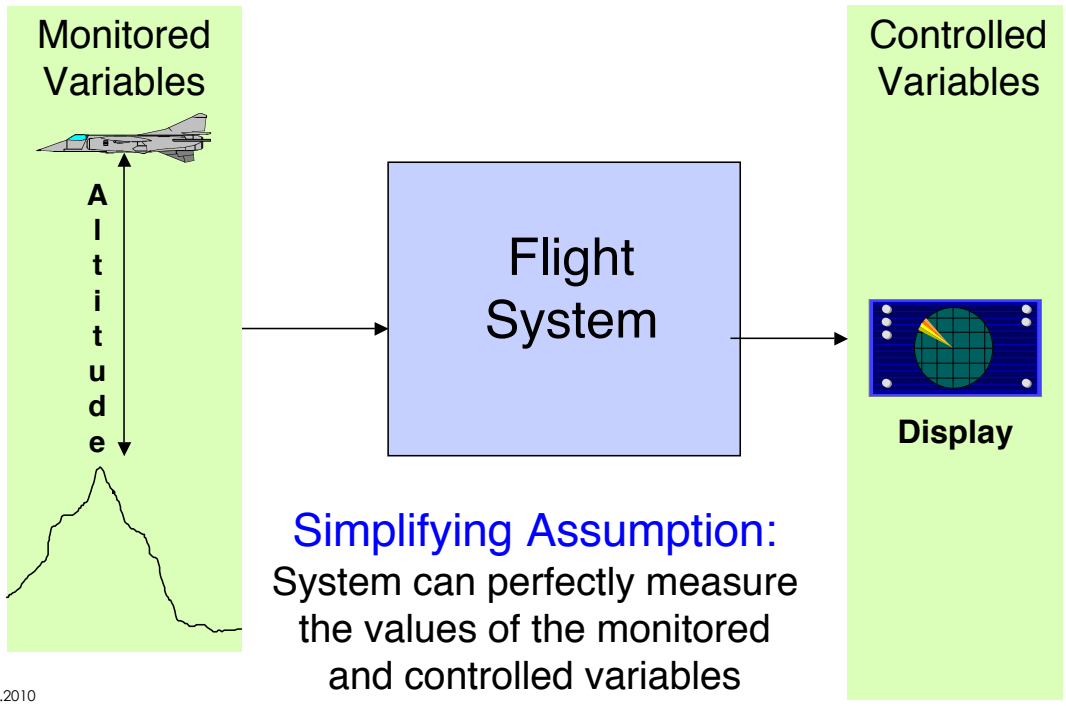
*D. Parnas and J. Madey, *Science of Computer Programming,* 1995.

MONITORED AND CONTROLLED VARIABLES

- To define the ideal system behavior, the requirements specification must
  - Identify and specify the controlled variables
  - Identify and specify the monitored variables
  - Specify the required relation betw. monitored and controlled vars
- Approach similar but not the same as Jackson's
  - **World** -> Environment
  - **Machine** -> Software (not System)



IDEAL SYSTEM BEHAVIOR



Simplifying Assumption:
System can perfectly measure the values of the monitored and controlled variables

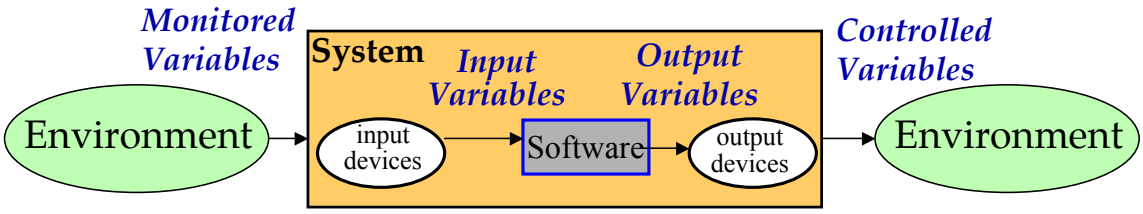
INPUT AND OUTPUT VARIABLES

- In specifying the real system behavior, the simplifying assumption is removed
- The values of input and output variables are read from input and output devices

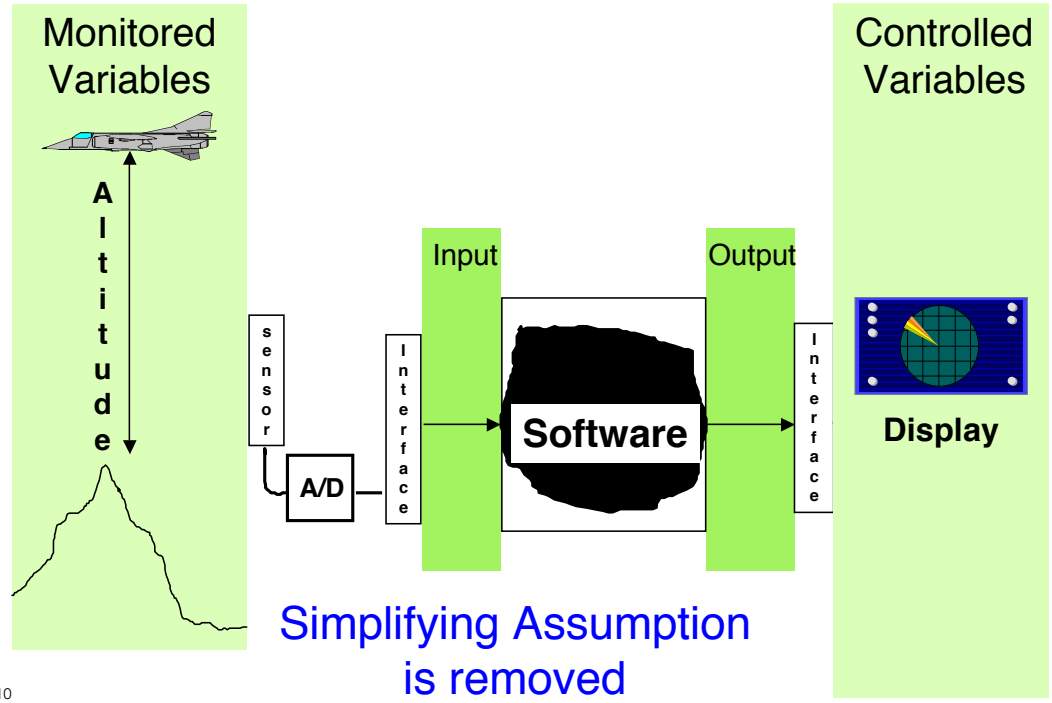
Other Two Variables of the FVM

(3) Inputs: variables from which the values of monitored variables may be estimated

(4) Outputs: variables available to affect controlled variables



REAL SYSTEM BEHAVIOR

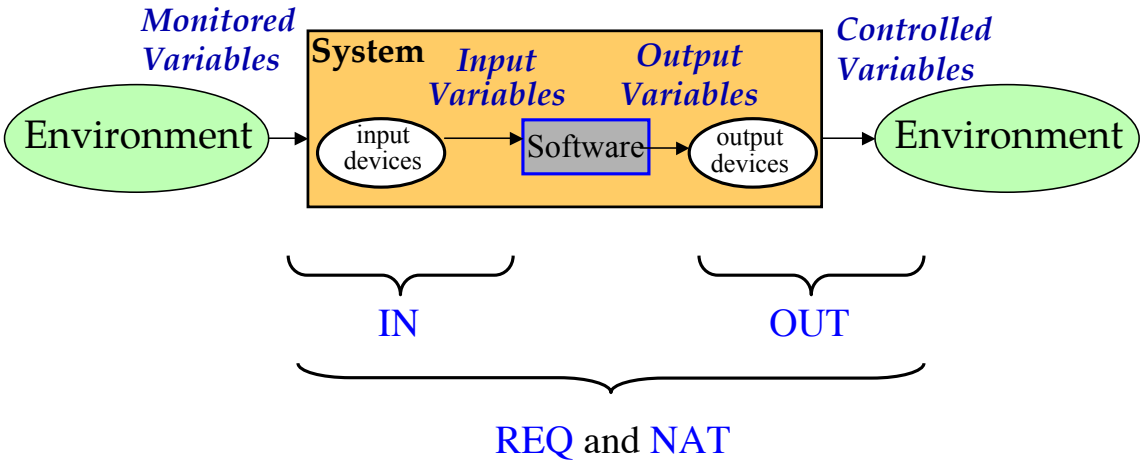


Simplifying Assumption is removed

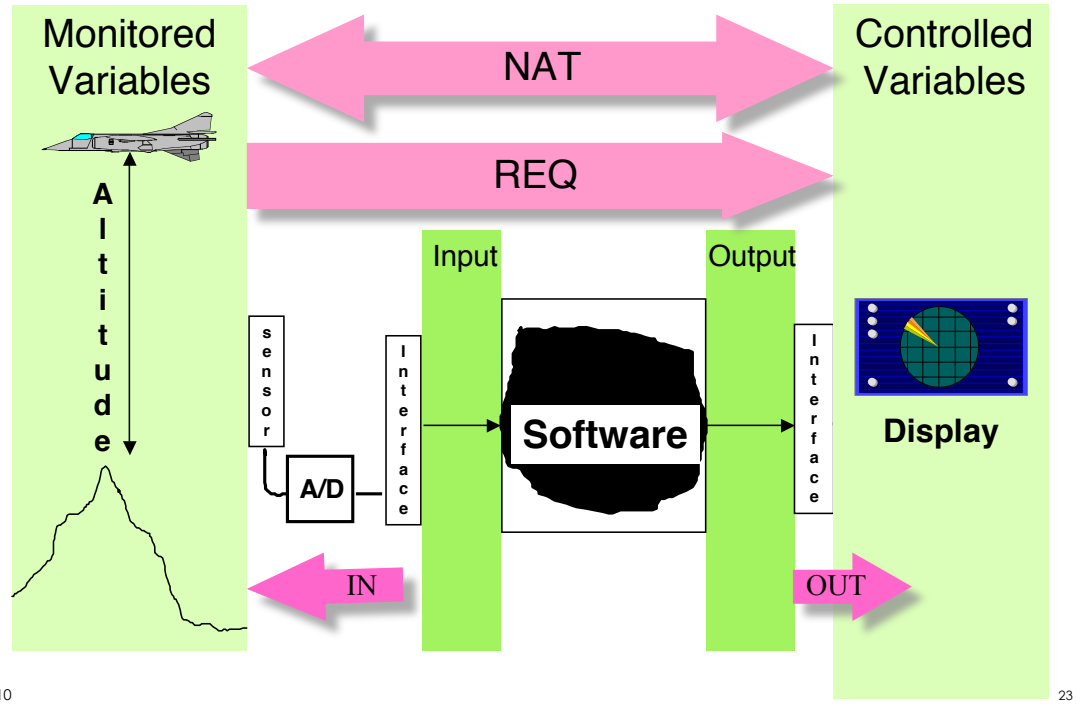
FOUR RELATIONS OF THE FOUR VARIABLE MODEL

- Two relations on monitored and controlled vars
  - NAT - Constraints imposed by the environment (e.g., physical laws, constraints of system environment)
  - REQ - Additional "constraints" system must impose on the environment to produce the required system behavior
- Two other relations
  - IN: Relation betw. monitored variables and inputs
  - OUT: Relation betw. outputs and controlled variables

FOUR VARIABLES AND FOUR RELATIONS OF THE FVM



THE FOUR VARIABLE VIEWPOINT: VARIABLES



SCR REQUIREMENTS MODEL: OVERVIEW

SCR: A "PRACTICAL" FORMAL METHOD

SCR was developed to address industry concerns while providing the benefits of a formal approach

- Four-Variable Model (Parnas 1991-1995)
- SCR Requirements Model (NRL 1992 - Present)
  - Special case of the Four Variable Model
  - Based on a state machine model
  - Explicit formal semantics
  - Major goal: Tool-based requirements method
    - » Formal basis for tool-assisted consistency checking, simulation, formal verification, etc.

SCR: Software Cost Reduction

SCR REQUIREMENTS MODEL (1)

state variables

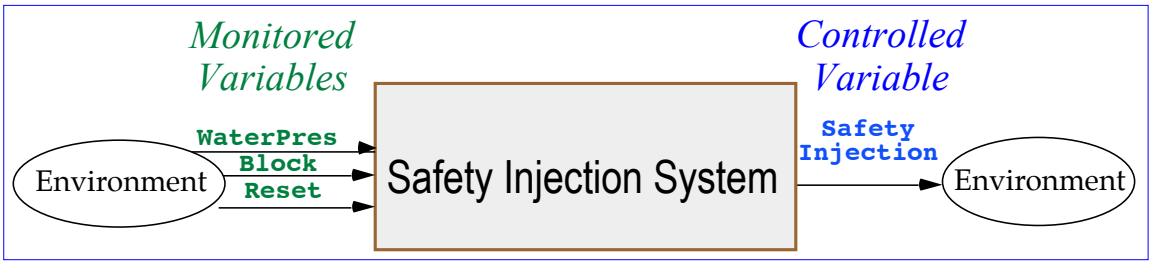
- Two classes of environmental variables
  - **Monitored variables**
  - **Controlled variables**
- Two classes of auxiliary variables
  - **Modes**: capture the history of values of monitored vars
  - **Terms**: functions on one or more state vars
- **REQ** specified by a set of functions defined on the state vars
- **NAT** includes a set of assumptions defined on the env vars

SCR REQUIREMENTS MODEL (2)

- Two classes of state predicates
  - **Condition**: a predicate on a single state
    Altitude > 500 or **Pressure = 500**
  - **Event**: a predicate on two states - denotes a change in state
    @T(Altitude > 500)
- Changes in monitored vars (monitored events) provide input alphabet of the state machine
  - Model is input-driven by monitored event
- The value of each *dependent state variable* (mode class, term, controlled var)
  - Is represented as a function in a table format
- The next state function of the state machine
  - Is the composition of these functions

EXAMPLE MODEL: CONTROL SYSTEM FOR SAFETY INJECTION (1)

- Based on a control system in a real nuclear power plant*
- System required to turn on safety injection when water pressure falls below a threshold 'Low'
- Operator can set a Block button to inhibit safety injection and a Reset button to reset the system after blockage

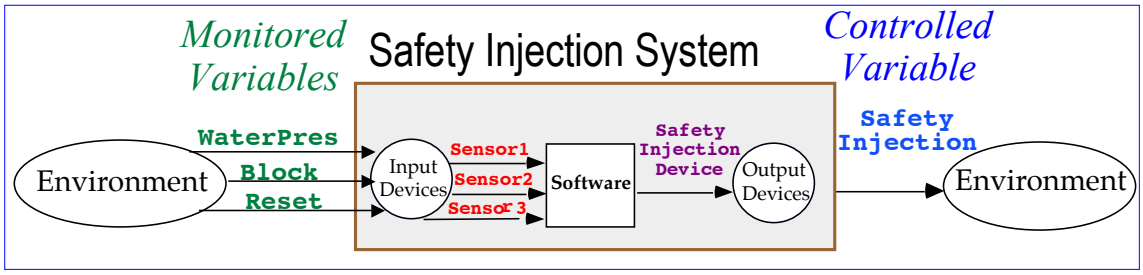


IDEAL SYSTEM BEHAVIOR

*Courtois,Parnas, "Documentation for Safety-Critical Software," *Proc., ICSE'93*, Baltimore

EXAMPLE MODEL: CONTROL SYSTEM FOR SAFETY INJECTION (2)

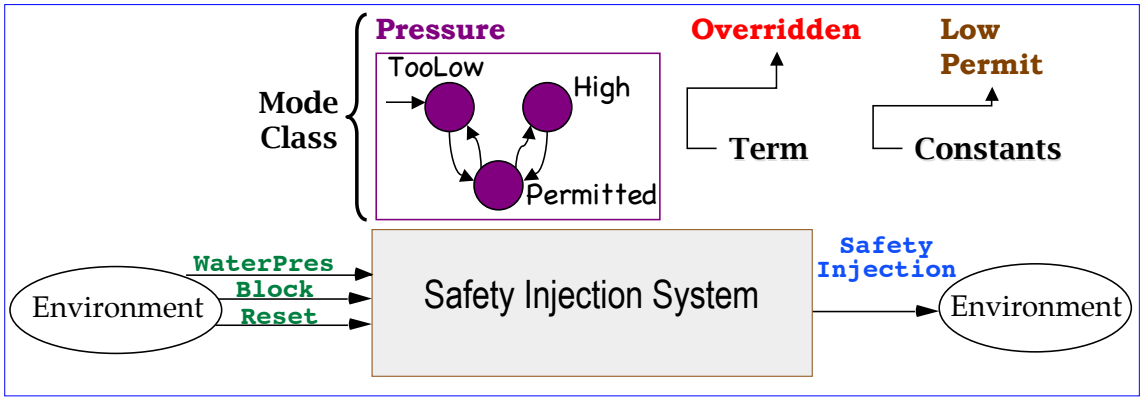
To estimate water pressure, the system uses three sensors



REAL SYSTEM BEHAVIOR

EXAMPLE MODEL: CONTROL SYSTEM FOR SAFETY INJECTION (3)

- Mode Class Pressure - abstraction of WaterPres
- Term Overridden - denotes whether operator has overridden injection
- Controlled variable SafetyInjection - defined in terms of terms, modes, and monitored variables



SUMMARY

- Specifying requirements precisely probably the most difficult aspect of software dev'ment
  - Good requirements specs in industry are very rare
- Needed: "Practical" formal methods for specifying/analyzing requirements
- To design for ease of change, requirements should be specified in at least two stages
  - Normal (Ideal) System Behavior first
  - Real Behavior later